القوائم Lists

2. الطابورQueue:

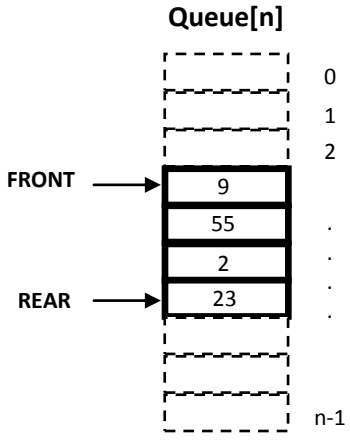
1.2 التعريف Definition :

- الطابــور queue هو قائمــة خطيــة linear list من العناصر items ، يستخــدم في مؤشرين two pointers :
  - المؤشر الأول The first pointer : يدعي المؤشر الخلفــي REAR يستخدم لإدخال insert عنصر واحد في الطابور queue .
  - المؤشر الثاني The second pointer : يدعي المؤشر الأمامــي FRONT يستخدم لإلغاء delete عنصر واحد من الطابور queue .

لذلك، الطابور queue يعتبر قائمة متغيرة Dynamic List .

- الطابور queue يعتبر First-In-First-Out (FIFO) data structure : العنصر الأول المضاف للطابور queue هو أول عنصر يتم إزالته من الطابور queue . وهذا يكون مكافئ لمتطلبات تلك العناصر ، كل العناصر التي يتم إضافتها أولاً يتم إزالتها أولاً.

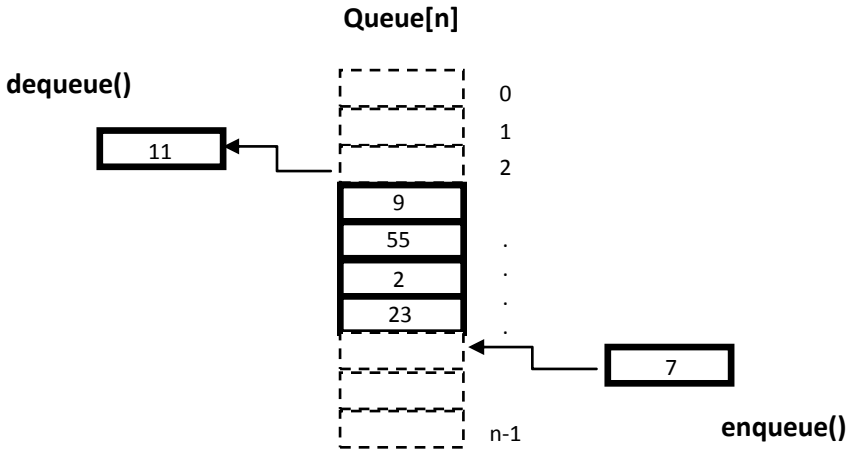
FIFO → (First in, First out)



2.2 العمليات التي تجرى على الطابور The operations of queue :

العمليات الرئيسية التي تنجز على الطابور queue :

1. enqueue(item) : إدخال insert عنصر في الطابور queue من مؤشر rear .
2. dequeue() : إلغاء delete وإرجاع قيمة العنصر الذي يأشر له المؤشر front .



3.2 تطبيقات الطابور Application of Queue :

1. تقنية Round Robin technique لجدولة المعالج processor scheduling تكون مطبقة باستخدام تركيبة بيانات الطابور the queue data structure .
2. كل خدمات الزبائن مثل (نظام الحجز في الطيران) البرمجيات المصممة تستخدم تركيبة بيانات الطابور the queue data structure لتخزين معلومات الزبائن customers information .
3. Printer server routines تكون مطبقة باستخدام تركيبة بيانات الطابور the queue data structure .

4.2 أنواع الطابور Types of Queue :

الطابور queue يمكن أن يكون :

1. طابور خطي Linear queue .
2. طابور دائري Circular queue .

5.2 التطبيق للطابور Implementation of queue :

في أغلب لغات البرمجة العالية المستوى high level languages ، الطابور queue يمكن أن يطبق بسهولة.

```
CREATE  Q[n] , FRONT ← -1 , REAR ← -1
```

```
INSERT ( Q[n] , REAR , x )
IF  REAR = n-1 , THEN  "Queue Full"
IF  REAR = -1 & FRONT=-1 , THEN  FRONT ← 0
REAR  ←  REAR + 1
Q[REAR]  ←  x
END
```

```
DELETE ( Q[n] , FRONT, REAR )
IF  FRONT = REAR , THEN  FRONT = REAR =-1  "Queue Empty"
FRONT  ←  FRONT + 1
END
```

6.2 برنامج بلغة السي لتطبيق الطابور الخطي بواسطة المصفوفة Code with C language for implementing a linear queue with an array :

العمليات على الطابور Operations on Queue :

- **createqueue(q)** : لتكوين q كطابور فاضي queue empty .
- **enqueue(q,i)** : لإدخال عنصر i في q .
- **dequeue(q)** : للوصول وإزالة عنصر من الطابور q queue .
- **peek(q)** : للوصول إلى أول عنصر في الطابور q queue بدون إزالته.

تمثيل الطابور Queue Representation :

نستخدم الإعلانات التالية لتمثيل الطابور queue :

```
/*Define a queue of capacity 10*/
#define CAPACITY 10
typedef struct
{
    int front;
    int rear;
    int elements[CAPACITY];
} queue;
queue *q;
```

باستخدام هذه الإعلانات، نستطيع إنجاز العمليات المختلفة على الطابور queue .

تكوين الطابور الفاضي Creating an Empty Queue :

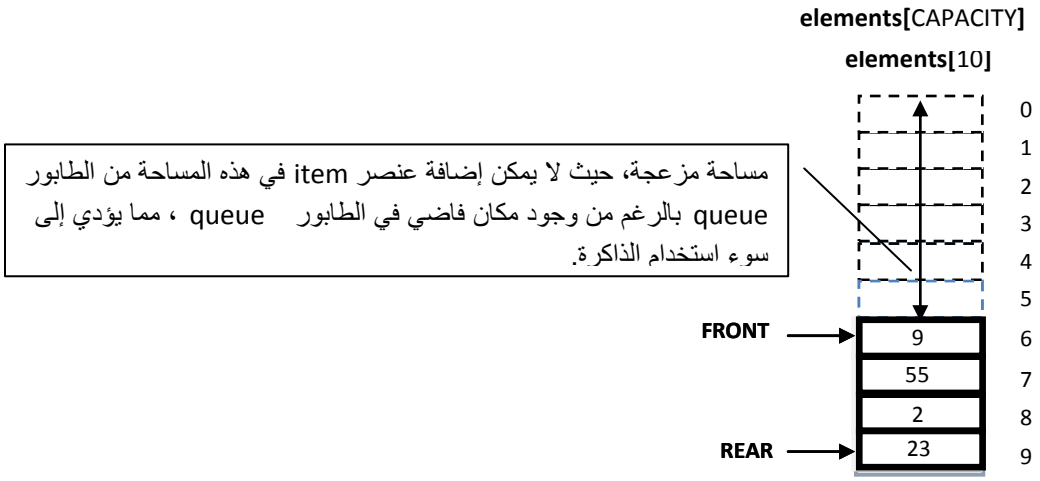
قبل أن نستخدم الطابور queue ، يجب أن يتم تكوينه. ويتم تكوين الطابور queue وذلك بتخصيص قيمة 1- لكل من المتغيرين front و rear . نلاحظ بأن المدى المسموح للمصفوفة هو من 0 إلى CAPACITY-1 .

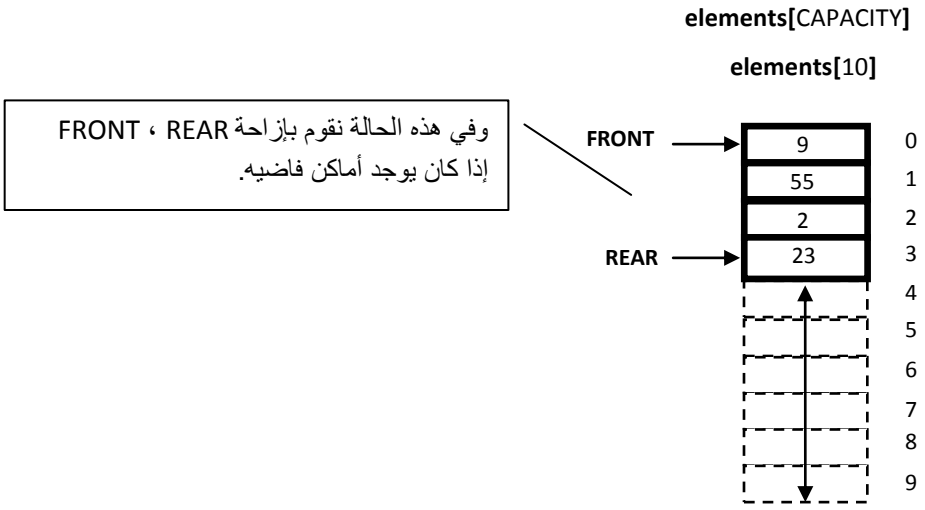
```
void createqueue(queue *q)
{
   q->front=q->rear=-1;
}
```

إنجاز عملية enqueue على الطابور الخطي Performing the enqueue Operation on a Linear Queue :

هناك سيناريوهان يجب أن ندرسهما، نفترض بأن الطابور queue غير ممتلئ not full :

1. إذا الطابور الخطي linear queue يكون فاضي empty ، إذا قيمة المتغيرين front و rear يكونان 1- ، إذاً قيمة المتغير rear تصبح 0 .
2. إذا الطابور الخطي linear queue غير فاضي not empty. إذاً هناك احتمالين آخرين:
   1. إذا قيمة المتغير rear أقل من CAPACITY - 1 ، إذاً قيمة المتغير rear تزداد بمقدار 1 .
   2. إذا قيمة المتغير rear تساوي CAPACITY – 1 ، إذا عناصر الطابور الخطي linear queue يتم تحريكها للا مام، والمتغيران front و rear يعدلان وفقاً لذلك.


elements[CAPACITY]
elements[10]
مساحة مزعجة، حيث لا يمكن إضافة عنصر item في هذه المساحة من الطابور queue بالرغم من وجود مكان فاضي في الطابور queue ، مما يؤدي إلى سوء استخدام الذاكرة.
FRONT
REAR
9
55
2
23
0
1
2
3
4
5
6
7
8
9


elements[CAPACITY]
elements[10]
وفي هذه الحالة نقوم بإزاحة FRONT ، REAR إذا كان يوجد أماكن فاضيه.
FRONT
REAR
9
55
2
23
0
1
2
3
4
5
6
7
8
9

```
void enqueue(queue *q, int value)
   {
     int i;
     if(q.front==-1 && q.rear==-1)          /* Queue is empty */
        q->front=0;
     else

      if ((q->rear == CAPACITY-1) && (q->front > 0))
       {
         for(i=q->front;i<=q->rear;i++)
            q->elements[i-q->front]=q->elements[i];
         q->rear=q->rear-q->front;
         q->front=0;
       }
      else

        if ((q->rear == CAPACITY-1) && (q->front == 0)) /*Queue is full*/
         {  printf("Queue is full");
            exit(0);
         }

     q->rear++;
     q->elements[q->rear]=value;
 }
```

فيضان الطابور Queue overflow : تحدث هذه الحالة نتيجة لمحاولة إضافة عنصر item إلى طابور ممتلئ full queue .

إنجاز عملية dequeue على الطابور الخطي Performing the dequeue Operation on a Linear Queue :

هناك احتمالين:

1. إذا هناك يكون عنصر واحد فقط في الطابور الخطي linear queue ، إذا بعد degueueing سيصبح الطابور الخطي فاضي empty. هذه الحالة للطابور الخطي linear queue تكون ظاهرة بوضع قيمة المتغيرين front و rear تساوي 1- .
2. القيمة للمتغير front تزداد بمقدار 1 .

```
void dequeue(queue *q)
 {
  if(q->front == q->rear)
    q->front=q->rear=-1;
   else
    q->front++;
}
```

نزوح الطابور Queue overflow : تحدث هذه الحالة نتيجة لمحاولة إزالة عنصر item من طابور فاضي empty queue .

الوصول إلى العنصر الأول في الطابور Accessing to the first Element :

هناك قد تكون حالات نريد فيها الوصول إلى العنصر الأول في الطابور queue بدون إزالته من الطابور queue .

```
int peek(queue *q)
{
    return(q->elements[q->front]);
}
```